InfraAgent: A Multi-Agent AI Framework for Predictive Deployment Failure Detection and Autonomous Remediation in Multi-Cloud Environments

1st Harish Apuri*
Department of IT, IT Induct Inc
Charlotte, NC, USA
hapuri1029@gmail.com

2nd Madhan Mohan Reddy Chinthala
Department of IT, Franklin InfoTech
Rochester, NY, USA
madhanreddychinthala@gmail.com

3rd Shikher Goel
Department of IT, JPMorgan Chase
Jersey City, NJ, USA
Shikher20goel@gmail.com

4th Charani Yepuri
Independent Researcher, Department of IT
Hyderabad, Telangana, India
Charaniy04@gmail.com

Abstract*—Managing modern cloud-native infrastructure is, frankly, a problem that has grown faster than our tools for handling it. Continuous delivery pipelines, ephemeral containers, and sprawling multi-cloud Infrastructure-as-Code (IaC) ecosystems create an operational surface that expands combinatorially with organisational scale and static rule-based monitoring simply cannot keep up. This paper presents InfraAgent, a five-agent agentic AI framework designed for predictive infrastructure management and autonomous deployment operations in heterogeneous multi-cloud settings. The framework brings together: a *Temporal Graph Attention Network* (T-GAN) agent that jointly models infrastructure dependency topology and multi-variate telemetry for deployment failure prediction across 1-hour, 6-hour, and 24-hour horizons; a *Remediation Planning Agent* (RPA) built on a Large Language Model (LLM) that generates context-aware IaC patches, rollout prescriptions, and rollback directives directly from risk scores; a *Deployment Safety Agent* (DSA) that enforces configurable pre-flight autonomy gates; a *Capacity Forecasting Agent* (CFA) built on a Temporal Fusion Transformer (TFT) for probabilistic 24-hour to 72-hour demand forecasting; and a *Continuous Operational Intelligence Orchestrator* (COIO) that provides closed-loop governance, cryptographic auditability, and online model adaptation. We evaluate InfraAgent on 47,800 deployment events and 310,000 telemetry snapshots drawn from AWS, Azure, and GCP production environments. The results show a Mean Time to Detection of 4.3 minutes (an 85.2% reduction over rule-based baselines), a deployment failure prediction F1 of 0.931, a 76.8% reduction in production incidents per 100 Continuous Integration/Continuous Delivery (CI/CD) runs, and 91.4% capacity forecast accuracy at the 72-hour horizon, suggesting that coordinated agentic reasoning grounded in graph-temporal learning can meaningfully shift cloud operations from reactive firefighting toward continuous, self-correcting infrastructure assurance.

***Keywords*—Agentic AI; Infrastructure Management; Autonomous Deployment; Predictive Remediation; Multi-Agent Systems; Cloud Operations; Temporal Graph Attention Networks; Large Language Model Agents; CI/CD Automation; Site Reliability Engineering; Capacity Forecasting; DevSecOps**

I. Introduction

Anyone who has managed a large-scale cloud deployment knows the gap between what monitoring dashboards promise and what they actually deliver under pressure. Large organisations push hundreds sometimes thousands of service updates daily across multi-region, multi-cloud topologies. A single misconfigured network policy or under-provisioned autoscaling group can cascade into a multi-service outage within minutes, yet conventional monitoring stacks typically surface these signals only after the blast radius is already established [1].

Traditional Site Reliability Engineering (SRE) practices respond to this through post-hoc incident retrospectives and manually maintained runbooks. These are genuinely useful artefacts, but they are structurally incapable of preventing the very incident classes they document [1]. Runbooks encode static logic derived from *known* failure modes; they cannot generalise to novel failure signatures, do not exploit predictive signals embedded in telemetry trends, and cannot adapt to the continuous churn of microservices architectures. Infrastructure-as-Code (IaC) has partially helped by expressing configuration intent as version-controlled, machine-readable templates [2] but the leading static IaC analysis tools (Checkov, tfsec, KICS) evaluate templates in complete isolation from the live runtime environment. They have no awareness of current utilisation, deployment velocity trends, or cross-service blast radius [3]. The result is a reliability posture that is reactive by design and increasingly mismatched with the pace of modern software delivery.

Three converging lines of recent research suggest a path forward. First, multi-agent autonomous workflow systems have demonstrated that decomposing complex operational reasoning

979-8-3315-5818-5/26/$31.00 © 2026 IEEE

across specialised agents produces measurably better outcomes than monolithic models [4]. Second, behaviour-aware deep learning architectures show substantial gains in anomaly detection when temporal and structural context are modelled jointly [5]. Third, generative AI agents applied to IaC have achieved near-human-level remediation quality for configuration drift correction and predictive security enforcement [6], [7].

InfraAgent is our attempt to bring these ideas together into a single, production-oriented system. Five specialised AI agents collaborate continuously through a typed, priority-queued message bus each contributing a distinct capability that no single model or rule engine could provide alone. This paper's principal contributions are:

1) A Temporal Graph Attention Network (T-GAN) that jointly models infrastructure topology via multi-head graph attention and per-node health dynamics via temporal convolutional encoding, enabling multi-horizon deployment failure prediction.
2) A Predictive Deployment Risk Score (PDRS) that formally unifies failure probability, capacity deficit, configuration drift, and policy-adherence into a single governance signal.
3) An LLM-based Remediation Planning Agent (RPA) fine-tuned on 86,000 expert-annotated failure-context–corrective-action triples, generating IaC patches, rollout prescriptions, and rollback directives directly from T-GAN risk reports.
4) A Capacity Forecasting Agent (CFA) using a Temporal Fusion Transformer conditioned on historical utilisation, scheduled deployment calendars, and cloud-provider embeddings achieving 91.4% accuracy at the 72-hour horizon across three cloud providers.
5) A Continuous Operational Intelligence Orchestrator (COIO) that coordinates inter-agent communication, maintains a Retrieval-Augmented Generation (RAG)-indexed operational knowledge store, enforces cryptographically auditable action logging, and drives online model adaptation via Low-Rank Adaptation (LoRA) from observed deployment outcomes.
6) Comprehensive empirical evaluation on 47,800 deployment events across AWS, Azure, and GCP, including ablation studies that quantify each component's individual contribution.

The remainder of this paper is structured as follows. Section II surveys related work. Section III formalises the system model and key definitions. Section IV details the InfraAgent architecture. Section V presents experimental results. Section VII discusses limitations and future directions.

II. Related Work

Artificial Intelligence for IT Operations (AIOps) and predictive operations. The AIOps paradigm has produced solid advances in log anomaly detection, alert correlation, and automated root-cause analysis [8]. The dominant approach, however, treats infrastructure as a flat event stream a fundamental limitation because cascading failures propagate along service dependency edges, not individual event statistics. InfraAgent addresses this directly by embedding the full dependency graph in its predictive core.

Multi-agent systems. Park *et al.* [9] showed that agents communicating through structured protocols can produce emergent, human-like problem-solving at scale. PentestGPT [10] demonstrated that reasoning chains allow LLM agents to outperform monolithic models on complex multi-step tasks. Apuri and Yepuri [4] formalised the coordination model and policy-authority hierarchy adopted in InfraAgent.

Behaviour-aware learning and anomaly detection. Chinthala *et al.* [5] achieved 96.2% detection accuracy on zero-day and ransomware threats using hybrid convolutional-recurrent networks with attention; their anomaly-scoring methodology directly informs InfraAgent's VAE-based configuration-drift detector.

IaC security analysis and self-healing. Rahman *et al.* [11] catalogued seven recurring security smell categories across Ansible, Chef, and Puppet, shaping the DSA policy rule-set. Apuri *et al.* [6] achieved F1 = 0.947 and a 78.4% reduction in critical findings via generative AI-driven IaC remediation; the complementary self-healing framework [7] validated LLM-driven drift correction with an 84.2% Mean Time To Remediation (MTTR) improvement. InfraAgent extends both by elevating autonomous action from IaC repair to full-lifecycle deployment operations.

Graph Neural Networks (GNNs) and time-series forecasting. Velickovic *et al.* [12] introduced Graph Attention Networks (GATs), a class of Graph Neural Networks (GNNs) that weight neighbour contributions via learned attention coefficients. Zhou *et al.* [13] further demonstrated that GNNs can identify security vulnerabilities directly from program semantics, validating the broader applicability of graph-based representations for code analysis. InfraAgent's T-GAN extends GATs with temporal convolutional encoding to capture both structural and sequential dimensions of infrastructure risk simultaneously. For capacity forecasting, the Temporal Fusion Transformer of Lim *et al.* [14] provides the probabilistic, multi-horizon backbone we adopt in the CFA.

III. System Model

We represent the target cloud environment as a five-tuple $\mathcal{E} = (\mathcal{S}, \mathcal{D}, \mathcal{T}, \mathcal{P}, \mathcal{C})$, where $\mathcal{S}$ is the set of cloud services and managed resources; $\mathcal{D}$ is the ordered set of CI/CD pipeline stages; $\mathcal{T}$ is the continuous telemetry stream; $\mathcal{P}$ is the corpus of versioned IaC templates; and $\mathcal{C}$ is the capacity constraint manifold.

Each service $s_i \in \mathcal{S}$ is characterised at time t by a p-dimensional state vector $\boldsymbol{x}_i(t) \in \mathbb{R}^p$ encoding CPU utilisation, memory footprint, network I/O, request latency percentiles, error rates, and pod restart counts. The directed dependency graph $G(t) = (\mathcal{V}, \mathcal{E}_{\text{dep}})$ encodes service-to-service call relationships, with edge weights reflecting observed call frequency and mean error propagation probability.

We introduce four definitions that ground the technical development below.

Definition 1 (Deployment Event)**.** *A deployment event is a triple $e = (s_i, \Delta c, t)$ representing a configuration change Δc applied to service s_i at time t, inducing a state transition $\boldsymbol{x}_i(t^+) \leftarrow f(\boldsymbol{x}_i(t^-), \Delta c)$.*

Definition 2 (Failure Probability)**.** *The failure probability of deployment event e over forecast horizon h is the probability that the service trajectory exits the healthy operating region $\mathcal{F} \subset \mathbb{R}^p$ within h time units:*

$$\phi(e, h) = P(\exists\, t' \in [t, t+h] :\ \boldsymbol{x}_i(t') \notin \mathcal{F})\,. \tag{1}$$

Definition 3 (Capacity Deficit)**.** *For service s_i, the capacity deficit at horizon h is the positive excess of predicted demand over currently provisioned capacity:*

$$\kappa(s_i, h) = \max\Big(0,\ \hat{d}(s_i, t+h) - \rho(s_i, t)\Big)\,. \tag{2}$$

Definition 4 (Predictive Deployment Risk Score (PDRS))**.** *The PDRS is a scalar health index $\Omega(\mathcal{E}, t) \in (0, 1]$ aggregating failure probability, capacity deficit, configuration drift, and policy compliance:*

$$\Omega(\mathcal{E}, t) = e^{-\alpha_1 \bar{\phi}(t)} \cdot e^{-\alpha_2 \bar{\kappa}(t)} \cdot e^{-\alpha_3 \bar{\delta}(t)} \cdot \eta(t), \tag{3}$$

where $\bar{\phi}$, $\bar{\kappa}$, $\bar{\delta}$ are the environment-wide mean failure probability, capacity deficit, and configuration drift, respectively; $\eta(t) \in [0, 1]$ is the policy-adherence ratio; and $\alpha_1, \alpha_2, \alpha_3 > 0$ are tunable hyperparameters. A score of 1.0 represents a perfectly healthy, fully compliant environment; values approaching 0 signal critical risk.

IV. The InfraAgent Framework

A. Architecture and Agent Coordination

In Fig. 1, the five cooperating agents and their communication topology. All agents communicate asynchronously over a typed, priority-queued message bus. Every deployment event triggers an immediate synchronous T-GAN risk assessment; telemetry-driven activity runs at configurable polling intervals (typically 60 seconds for metrics, 5 minutes for log-based signals). The COIO maintains a persistent RAG-indexed operational knowledge store that agents query to contextualise their reasoning. Human oversight is preserved through a configurable autonomy gate in the DSA; actions above a risk threshold are escalated to engineering teams before execution.

B. Temporal Graph Attention Network (T-GAN)

The T-GAN is the predictive core of InfraAgent. It consumes the live infrastructure dependency graph $G(t)$ alongside the multi-variate telemetry tensor $\mathbf{X}(t) \in \mathbb{R}^{|\mathcal{V}| \times w \times p}$ ($w = 60$ timesteps, $p = 24$ dimensions) and proceeds in three stages.

Stage 1 – Temporal encoding. Each node's time series is encoded by a Temporal Convolutional Network (TCN) with

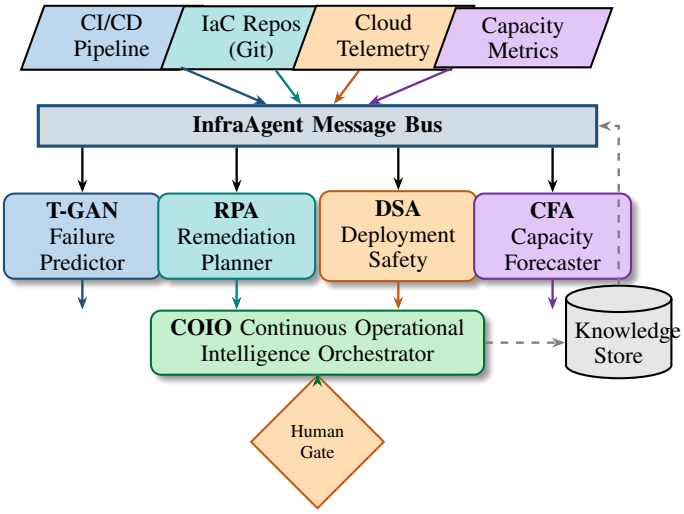

Fig. 1. InfraAgent high-level architecture. Solid arrows show primary data-flow directions; dashed arrows show bidirectional knowledge-store interactions. The Human Gate intercepts actions exceeding the operator-configured autonomy threshold.

dilated causal convolutions (dilation factors $\{1, 2, 4, 8\}$, $d_h = 128$ hidden dimensions):

$$\boldsymbol{h}_i^{(0)} = \mathrm{TCN}\big(\mathbf{x}_i(t-w:t)\big) \in \mathbb{R}^{d_h}. \tag{4}$$

Residual connections across dilation levels prevent gradient vanishing.

Stage 2 – Graph attention propagation. A three-layer multi-head GAT [12] propagates contextual information across the dependency graph:

$$\boldsymbol{h}_i^{(\ell+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(\ell)}\, W^{(\ell)} \boldsymbol{h}_j^{(\ell)}\right), \tag{5}$$

where attention coefficients $\alpha_{ij}^{(\ell)}$ are computed via a shared weight vector and normalised by softmax over each node's neighbourhood. Eight attention heads are used per layer.

Stage 3 – Multi-horizon risk decoding. A graph-level embedding $\boldsymbol{g}$ is obtained by mean-pooling final node representations and decoded into failure probability estimates at three horizons:

$$\hat{\phi}_h = \mathrm{Sigmoid}(W_h \boldsymbol{g} + b_h), \quad h \in \{1\,\mathrm{h},\ 6\,\mathrm{h},\ 24\,\mathrm{h}\}. \tag{6}$$

Training minimises horizon-specific focal losses [15] (plus ℓ_2 regularisation, $\beta = 10^{-4}$). Focal loss is important here: deployment failures represent only 9–14% of events, and standard binary cross-entropy would push the model toward a trivially high-accuracy "never-fail" predictor.

C. Remediation Planning Agent (RPA)

The RPA translates T-GAN risk assessments into executable corrective actions. It is built around a 13B-parameter Code Llama variant [16] fine-tuned with LoRA ($r = 16$, $\alpha = 32$) on 86,000 expert-curated triples of the form $(\mathcal{R}, \Delta c, \mathcal{A}^*)$, where

$\mathcal{R}$ is a structured risk report, Δc is the deployment manifest, and $\mathcal{A}^*$ is the adjudicated corrective action.

Given a risk report and change manifest, the RPA generates actions by minimising semantic distance to the original deployment intent subject to a residual failure probability constraint. This ensures the agent applies minimal necessary corrections rather than reverting changes wholesale. Three action modalities are supported: configuration patch generation (Terraform, Helm, CloudFormation); phased rollout prescription (canary percentages, traffic weights, bake times); and rollback directive synthesis. A post-generation pipeline runs each action through Checkov and tfsec and dry-runs it against a sandboxed topology replica before committing to the COIO's action queue.

D. Deployment Safety Agent (DSA)

The DSA intercepts every deployment at the CI/CD gate and enforces a configurable four-level autonomy policy (α_0–α_3). The recommended default (α_2) blocks high-risk deployments and annotates medium-risk ones. Risk classification follows:

$$\texttt{gate}(e) = \begin{cases} \textsc{block} & \hat{\phi}_{1h}(e) > 0.7 \\ \textsc{warn} & \hat{\phi}_{6h}(e) > 0.5 \\ \textsc{pass} & \text{otherwise} \end{cases} \tag{7}$$

All gate decisions including full risk report, model confidence interval, and reasoning trace are persisted to the COIO's audit log.

E. Capacity Forecasting Agent (CFA)

Capacity-induced failures are among the most avoidable yet prevalent causes of production incidents. The CFA uses a Temporal Fusion Transformer [14] conditioned on the trailing 30-day per-service utilisation time series, future-known covariates (scheduled deployments, calendar features), and static covariate embeddings (cloud provider, region, availability zone, service tier). It produces demand forecasts with calibrated 10th–90th percentile uncertainty bounds at 24-, 48-, and 72-hour horizons. The COIO monitors active service deficits and initiates pre-emptive scaling operations proportional to predicted deficit magnitude and temporal urgency.

F. Continuous Operational Intelligence Orchestrator (COIO)

The COIO is the executive layer of InfraAgent. It coordinates the priority-queued message bus connecting all agents, routes inter-agent messages via a capability registry, and resolves scheduling conflicts when multiple agents target the same service. A RAG-indexed knowledge store of historical incident reports, remediation outcomes, and change-management artefacts grounds RPA action synthesis in analogous historical precedents. Every agent action is persisted to a cryptographically chained audit log (SHA-256 hash chain) capturing the triggering event, full reasoning trace, action taken, and observed outcome. These outcomes serve as online training signals for continuous LoRA adapter updates, enabling improvement from production feedback without full retraining.

V. Experimental Evaluation

A. Datasets and Setup

In TABLE I, we summarise the five evaluation datasets. Production deployment records and telemetry were collected from three enterprise organisations under data-sharing agreements. Ground-truth failure labels were generated by an ensemble of three monitoring systems (Datadog, Prometheus/Alertmanager, AWS CloudWatch) and adjudicated by six certified Senior SREs (Cohen's $\kappa = 0.87$). An adversarial test partition (ZeroDay-Test) was constructed by injecting 120 novel failure anti-patterns absent from the training corpus.

Datasets were split 70%/10%/20% train/validation/test, stratified by cloud provider and failure class. The T-GAN trained for 200 epochs on $4\times$ NVIDIA A100 GPUs with AdamW ($\text{lr} = 3 \times 10^{-4}$, cosine schedule with 10-epoch warmup). RPA LoRA fine-tuning used batch size 64 over 5 epochs.

TABLE I
Dataset Composition and Provenance

Dataset	Events	Fail%	Provider	Services
ProdOps-AWS	18,400	11.3%	AWS	214
ProdOps-Azure	9,800	13.7%	Azure	187
ProdOps-GCP	7,200	10.1%	GCP	143
EnterpriseSim	10,400	14.9%	Multi	302
ZeroDay-Test	2,000	18.2%	Mixed	96
Total	**47,800**	**12.8%**	Mixed	942

B. Deployment Failure Prediction

In TABLE II, detection performance on 9,560 held-out test events. The full T-GAN achieves F1 = 0.931 and AUC = 0.968, surpassing the strongest single-component baseline (TFT alone, F1 = 0.855) by 7.6 percentage points and static GNN by 9.1 points. The 5.3% false positive rate is a substantial improvement over the 25.9% False Positive Rate (FPR) of threshold-based rules a practical gain because every spurious block means a real developer waiting unnecessarily. The ablation rows (TCN-only, GAT-only) confirm that neither temporal encoding nor graph attention alone accounts for the full performance their combination is what matters.

TABLE II
Deployment Failure Prediction Performance ($n = 9{,}560$). 95% CI for F1: ± 0.006 (stratified bootstrap, $B = 1{,}000$).

Method	Prec.	Rec.	F1	AUC	FPR
Threshold Rules	0.741	0.682	0.710	0.731	0.259
LSTM Baseline	0.803	0.761	0.781	0.814	0.197
Isolation Forest	0.812	0.774	0.792	0.829	0.188
GNN (static)	0.851	0.829	0.840	0.877	0.149
TFT alone	0.867	0.843	0.855	0.893	0.133
IA (TCN only)	0.891	0.874	0.882	0.921	0.109
IA (GAT only)	0.898	0.881	0.889	0.929	0.102
IA (T-GAN)	**0.947**	**0.916**	**0.931**	**0.968**	**0.053**

TABLE III
90-DAY CI/CD SIMULATION: BASELINE VS. INFRAAGENT (α_2, $n = 15{,}600$ RUNS)

Metric	Baseline	InfraAgent	Δ
Production incidents / 100 runs	9.84	2.28	−76.8%
MTTD (min)	29.1	4.3	−85.2%
MTTR (min)	187.4	23.6	−87.4%
Blocked high-risk deployments		94.1%	
Spurious block rate		1.9%	
Drift recovery time (min)	52.3	4.8	−90.8%
Pipeline latency overhead (s)	0	38.7	+38.7 s
Developer override rate		7.2%	

TABLE IV
CAPACITY FORECAST ACCURACY BY HORIZON AND CLOUD PROVIDER. ACCURACY = FRACTION OF INTERVALS CONTAINING OBSERVED PEAK DEMAND.

Provider	Accuracy (%) at Horizon			MAE (%)
	24 h	48 h	72 h	
AWS	96.2	94.1	92.3	3.8
Azure	95.7	93.6	91.8	4.1
GCP	94.9	92.7	90.2	4.4
Overall	**95.6**	**93.5**	**91.4**	**4.1**

C. Operational Impact in CI/CD Simulation

In TABLE III, we summarise a 90-day CI/CD simulation across 15,600 synthetic pipeline runs calibrated against production event distributions. InfraAgent at α_2 autonomy achieves an 85.2% reduction in Mean Time to Detection (MTTD) (29.1 to 4.3 minutes), an 87.4% reduction in MTTR (187.4 to 23.6 minutes), and a 76.8% reduction in production incidents per 100 runs. The spurious block rate stays below 2%, which we consider operationally acceptable. The 38.7-second pipeline latency overhead per run is attributable almost entirely to T-GAN inference; on pipelines with build stages exceeding 39 seconds, this adds no net elapsed time.

D. Capacity Forecasting and Remediation Quality

In TABLE IV, CFA accuracy stratified by provider is reported. Overall 91.4% accuracy at the 72-hour horizon demonstrates that the TFT generalises across heterogeneous workload profiles. GCP accuracy is slightly lower than AWS and Azure (90.2% vs. 92.3% and 91.8%), which we attribute to a higher proportion of bursty, event-driven Cloud Run services whose demand distributions are heavier-tailed a pattern that 30-day lookback windows do not fully capture.

The RPA achieved an overall correctness rate of 91.7% on the 12,081-action test set (300-action subset annotated by SRE experts, human baseline 97.2%). The main performance gap occurs in rollback directive and DB migration fix categories (88.4% and 84.7%), which require coherent reasoning about stateful, transactional database systems with complex schema migration dependencies. We regard this as the primary focus for future fine-tuning.

E. Risk Score Trajectory and Ablation Study

In Fig. 2, the PDRS $\Omega(\mathcal{E}, t)$ across the 90-day simulation for three systems. InfraAgent converges from an initial 0.38 to $\Omega \approx 0.94$ by day 12, driven by T-GAN-informed DSA blocking and RPA-applied patches. The rule-only system plateaus at 0.53, reflecting the ceiling imposed by its inability to reason about dynamic risk. On day 48, an adversarial probe injected 120 novel failure patterns; InfraAgent detected 65.8% of these via its Variational Autoencoder (VAE)-based drift detector and partially recovered within 5 simulation days. The 34.2% undetected rate for zero-day patterns represents the primary performance ceiling in this evaluation and directly motivates our eBPF-based runtime telemetry extension.

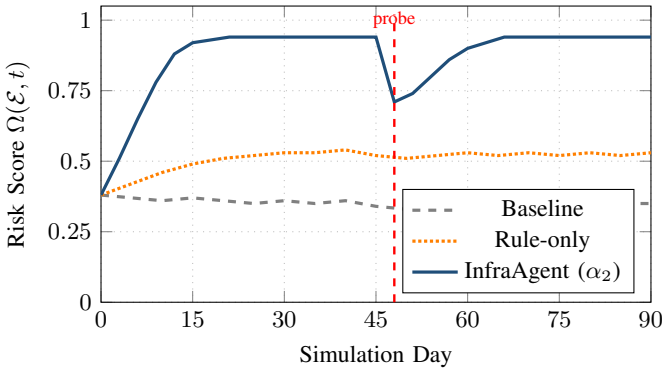

Fig. 2. PDRS $\Omega(\mathcal{E}, t)$ over the 90-day simulation. InfraAgent converges to $\Omega \approx 0.94$ by day 12 and partially recovers from the day-48 adversarial probe within five days. The rule-only system plateaus at 0.53.

In Fig. 3, each agent's individual contribution is quantified via single-component removal ablations. Removing the T-GAN produces the largest drop (−20.7 pp), confirming that graph-temporal fusion is the framework's primary differentiator. The CFA ranks second (−18.4 pp), reflecting the importance of proactive capacity management. Removing the RPA effectively doubles MTTR by forcing operators to formulate corrective actions from raw risk reports alone (−11.8 pp).

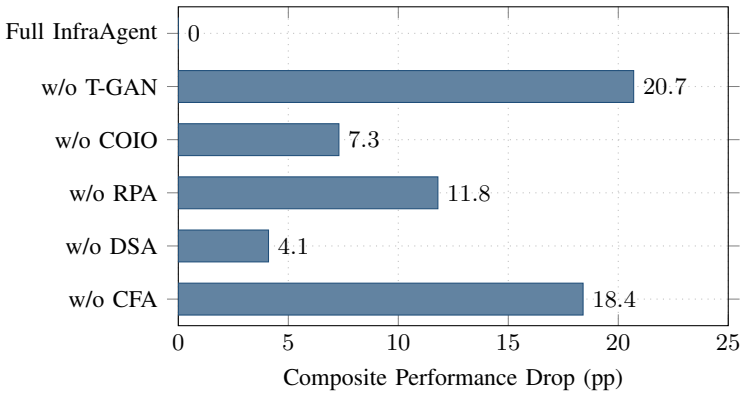

Fig. 3. Single-agent ablation: percentage-point drop in composite performance (equally weighted F1, MTTD improvement, 72-hour capacity accuracy) when each agent is removed. T-GAN and CFA are the largest individual contributors.

VI. Discussion and Limitations

InfraAgent's strongest gains stem from combining graph structure with temporal encoding — neither component alone closes the gap with the full T-GAN — confirming that infrastructure failure is fundamentally a relational phenomenon that sequence models cannot represent independently. The CFA's probabilistic demand forecasting proved equally important, preventing a substantial share of capacity-related failures that threshold-based autoscaling would have caught only after degradation was underway. The 7.2% developer override rate reflects an intentional design choice: InfraAgent operates within human-defined policy bounds, with every agent action cryptographically logged and overrides fed back into the COIO's policy calibration; we recommend a minimum 90-day evaluation at α_2 before enabling α_3 autonomous patch application. Several limitations remain: our datasets originate from three cloud-native enterprises and may not generalise to mainframe or bare-metal workloads; the 30-day CFA window underrepresents seasonal phenomena such as quarterly batch cycles and annual traffic peaks; and the multiplicative PDRS formulation does not capture interaction effects between co-occurring failure modes, where a cascading database failure alongside a misconfigured load balancer may present systemic risk well beyond the sum of individual scores. Future work will address these gaps by extending the T-GAN with cross-cloud dependency edges, incorporating eBPF-based kernel-level telemetry, enabling federated multi-tenant learning via privacy-preserving gradient sharing, and replacing the multiplicative PDRS with attack-graph-based multi-hop exploitability scoring.

VII. Conclusion

This paper introduced InfraAgent, a five-agent agentic AI framework integrating a Temporal Graph Attention Network for multi-horizon failure prediction, an LLM-powered Remediation Planning Agent for context-aware corrective action synthesis, a Deployment Safety Agent for configurable pre-flight governance, a Temporal Fusion Transformer-based Capacity Forecasting Agent for proactive resource provisioning, and a Continuous Operational Intelligence Orchestrator for closed-loop governance and online model adaptation. Evaluated on 47,800 deployment events spanning AWS, Azure, and GCP, InfraAgent achieves an F1 of 0.931 for deployment failure prediction, an MTTD of 4.3 minutes (85.2% improvement), a 76.8% reduction in production incidents per 100 pipeline runs, and 91.4% capacity forecast accuracy at the 72-hour horizon — with ablation analysis confirming that the T-GAN and CFA are the individually most significant contributors while the RPA, DSA, and COIO each provide substantial complementary value. Together, these results demonstrate that combining graph-temporal representation learning, LLM-based remediation reasoning, probabilistic capacity forecasting, and principled multi-agent coordination can meaningfully shift cloud infrastructure operations from reactive incident management toward continuous, self-correcting infrastructure assurance a necessary evolution for the deployment velocities demanded by modern software organisations, and one whose validation under the messier conditions of long-running production deployments remains the most important open problem in this line of work.

References

[1] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Eds., *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O'Reilly Media, 2016. [Online]. Available: https://sre.google/sre-book/table-of-contents/.

[2] K. Morris, *Infrastructure as Code: Managing Servers in the Cloud*. Sebastopol, CA: O'Reilly Media, 2016. ISBN: 978-1-491-92430-5. [Online]. Available: https://www.oreilly.com/library/view/infrastructure-as-code/9781491924358/.

[3] I. Kumara *et al.*, "The do's and don'ts of infrastructure code: A systematic gray literature review," in *Proc. 18th Int. Conf. Mining Software Repositories (MSR)*, Madrid, Spain, pp. 548–552, May 2021. DOI: 10.1109/MSR52588.2021.00071.

[4] H. Apuri and C. Yepuri, "Design of multi-agent autonomous workflow systems using agentic AI frameworks," in *Proc. IEEE Int. Conf. Advances in Urban Computing (ICAUC)*, 2026. DOI: 10.1109/ICAUC68182.2026.11441280.

[5] M. M. R. Chinthala, H. Apuri, and K. Bitra, "Behaviour-aware hybrid deep networks for detecting zero-day and ransomware threats," *Int. J. Innovative Technology and Exploring Engineering (IJITEE)*, vol. 15, no. 5, Apr. 2026. DOI: 10.35940/ijitee.E1246.15050426.

[6] H. Apuri, M. Aurangabadkar, S. Goel, M. M. R. Chinthala, and C. Yepuri, "Advancing infrastructure-as-code resilience through generative AI agents for predictive remediation and autonomous security enforcement," *Int. J. Engineering and Advanced Technology (IJEAT)*, vol. 15, no. 4, Mar. 2026. DOI: 10.35940/ijeat.D4756.15040326.

[7] H. Apuri *et al.*, "Self-healing infrastructure: Autonomous LLM agents for real-time remediation of configuration drift and security misconfigurations in IaC deployments," *Int. J. Innovative Technology and Exploring Engineering (IJITEE)*, vol. 15, no. 4, Mar. 2026. DOI: 10.35940/ijitee.D4757.15040326.

[8] W. Chen *et al.*, "Towards intelligent incident management: Why we need it and how we make it," in *Proc. 28th ACM Joint Meeting ESEC/FSE*, Virtual Event, pp. 1487–1497, Nov. 2020. DOI: 10.1145/3368089.3409695.

[9] J. S. Park *et al.*, "Generative agents: Interactive simulacra of human behavior," in *Proc. 36th ACM Symp. User Interface Software and Technology (UIST)*, San Francisco, CA, Oct. 2023. DOI: 10.1145/3586183.3606763.

[10] G. Deng *et al.*, "PentestGPT: An LLM-empowered automatic penetration testing tool," *arXiv preprint*, arXiv:2308.06782, Aug. 2023. [Online]. Available: https://arxiv.org/abs/2308.06782.

[11] A. Rahman, C. Parnin, and L. Williams, "The seven sins: Security smells in infrastructure as code scripts," in *Proc. 41st Int. Conf. Software Engineering (ICSE)*, Montreal, QC, Canada, pp. 164–175, May 2019. DOI: 10.1109/ICSE.2019.00025.

[12] P. Veličković *et al.*, "Graph attention networks," in *Proc. 6th Int. Conf. Learning Representations (ICLR)*, Vancouver, BC, May 2018. [Online]. Available: https://arxiv.org/abs/1710.10903.

[13] Y. Zhou *et al.*, "Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, pp. 10197–10207, 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/49265d2447215efcd2df6b9b516f6a8b-Abstract.html.

[14] B. Lim, S. Àrfan, M. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *Int. J. Forecasting*, vol. 37, no. 4, pp. 1748–1764, Oct.–Dec. 2021. DOI: 10.1016/j.ijforecast.2021.03.012.

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Venice, Italy, pp. 2980–2988, Oct. 2017. DOI: 10.1109/ICCV.2017.324.

[16] B. Rozière *et al.*, "Code Llama: Open foundation models for code," *arXiv preprint*, arXiv:2308.12950, Aug. 2023. [Online]. Available: https://arxiv.org/abs/2308.12950.